Forte™ for Java™ (Community Edition) QuickStart Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-4569-10
Revision 01, February 2000

Forte for Java (Community Edition) QuickStart Guide

Please
Recycle

February 2000

Contents

Getting Started With Forte for Java

Forte™ for Java™, Community Edition, provides a complete set of tools, integrated into a single environment, for developing cross-platform applications and applets written in the Java programming language. These tools enable you to edit, compile, debug, browse, and deploy Java programs.

In addition, Forte for Java makes it easy for you to design a graphical user interface (GUI). You can choose components (such as windows, dialog boxes, and buttons) in the component palette and place them in the Form Editor, where you can lay them out. All changes made to the graphical interface (for example, adding a button) are automatically reflected in the source code.

Forte for Java also makes it easy for you to create connections between components. For example, you can create a connection between a dialog box and a button so that the dialog box opens when the user clicks the button. Forte for Java automatically generates the Java code for the action you specify.

Looking at the Forte for Java Environment

When you first start Forte for Java, several windows appear, as shown in FIGURE 1. The *main window* provides access to windows, menus, and toolbars that you can use to develop a Java applet or application. Of particular interest is the *component palette*, where you can access components for building your GUI.

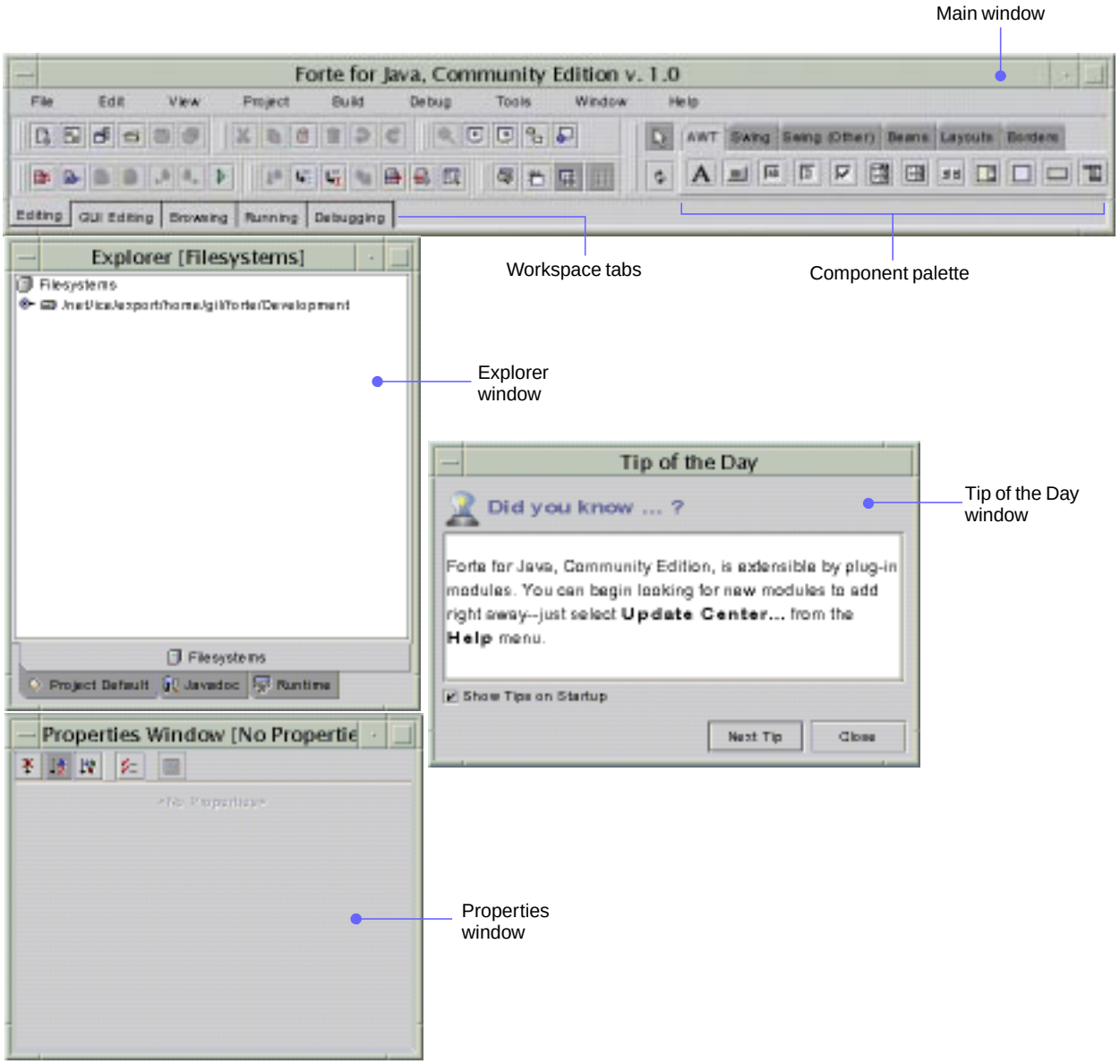
The *Explorer window* provides a hierarchical view of the packages, objects, and files in the Forte for Java environment. The Explorer window is a good starting point for working with the different parts of your application.

The Properties window enables you to view and edit the properties of your components. For example, you can use this window to assign text to a button and to change its color, size, and font.

The Tip of the Day window provides information on how to use the features in Forte for Java more effectively. If you feel comfortable using the environment, you can turn off the display of this window at startup.

The Forte for Java environment includes five workspaces to help you work more efficiently. Each *workspace* consists of a set of windows geared toward a specific task. These tasks are editing, GUI editing, browsing, running, and debugging. You can switch from one workspace to another by clicking its tab in the main window.

FIGURE 1 Forte for Java at Start-up

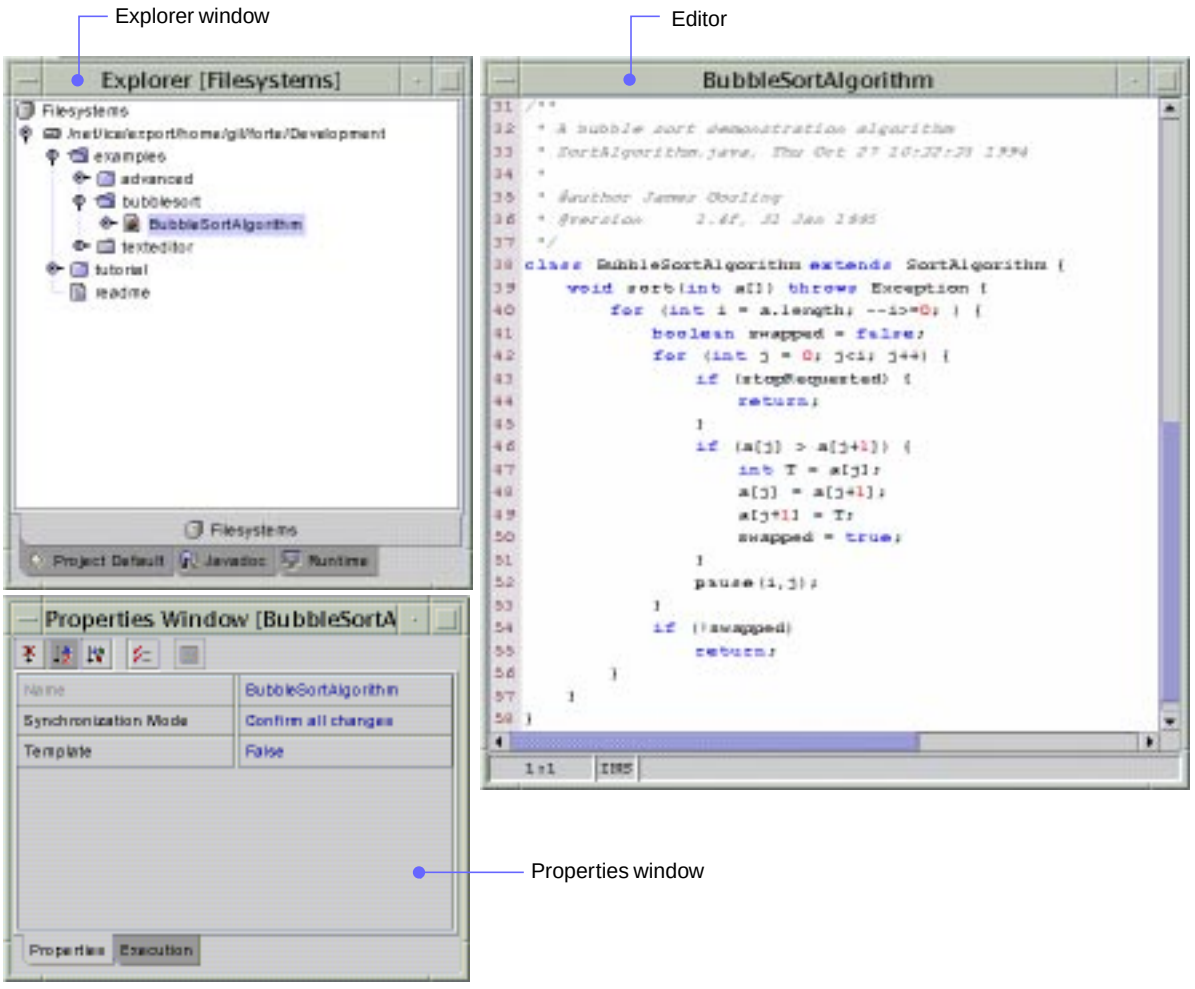


Your current workspace does not constrain the windows you can have open. You can use the View menu on the main window to open any window at any time. When you exit Forte for Java, it saves the state of each workspace. The next time you launch the program, the windows in your workspaces appear exactly as you left them.

Editing Workspace

When you first start Forte for Java, you are in the editing workspace, which contains the Explorer and Properties windows. When you open a file, the *Editor* appears, which is the tool for editing Java, HTML, and plain text files. In the Editor, the source code is syntactically colored—default keywords, for example, are in blue. The Editor also supports *dynamic code completion*—you can type the first few characters of an expression and then view a list of classes, methods, and variables that can be used to complete the expression.The following figure shows the Explorer window, Properties window, and Editor.

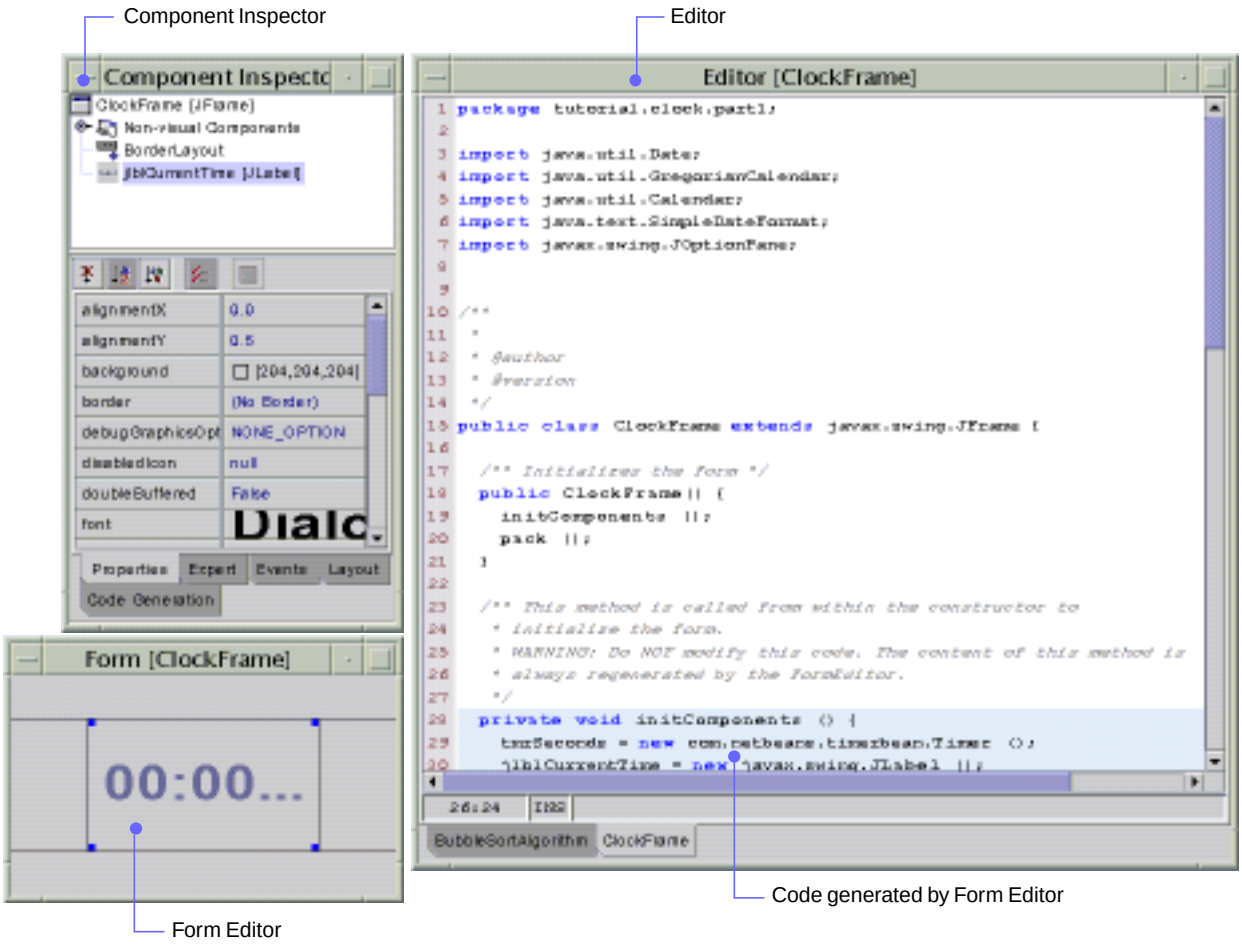
FIGURE 2 Editing Workspace



GUI Editing Workspace

The GUI editing workspace is the main area in which you develop a graphical user interface. This workspace includes the Explorer and Editor (as in the editing workspace) plus two additional windows. The *Component Inspector* enables you to view the components in your application and set their properties. The *Form Editor window* is the primary area for creating and modifying a graphical interface. Code generated by the Form Editor appears with a shaded background in the source Editor and can not be edited manually. If you open the Form Editor window in another workspace, Forte for Java automatically switches to the GUI editing workspace. The following figure shows the Component Inspector, Form Editor, and Editor.

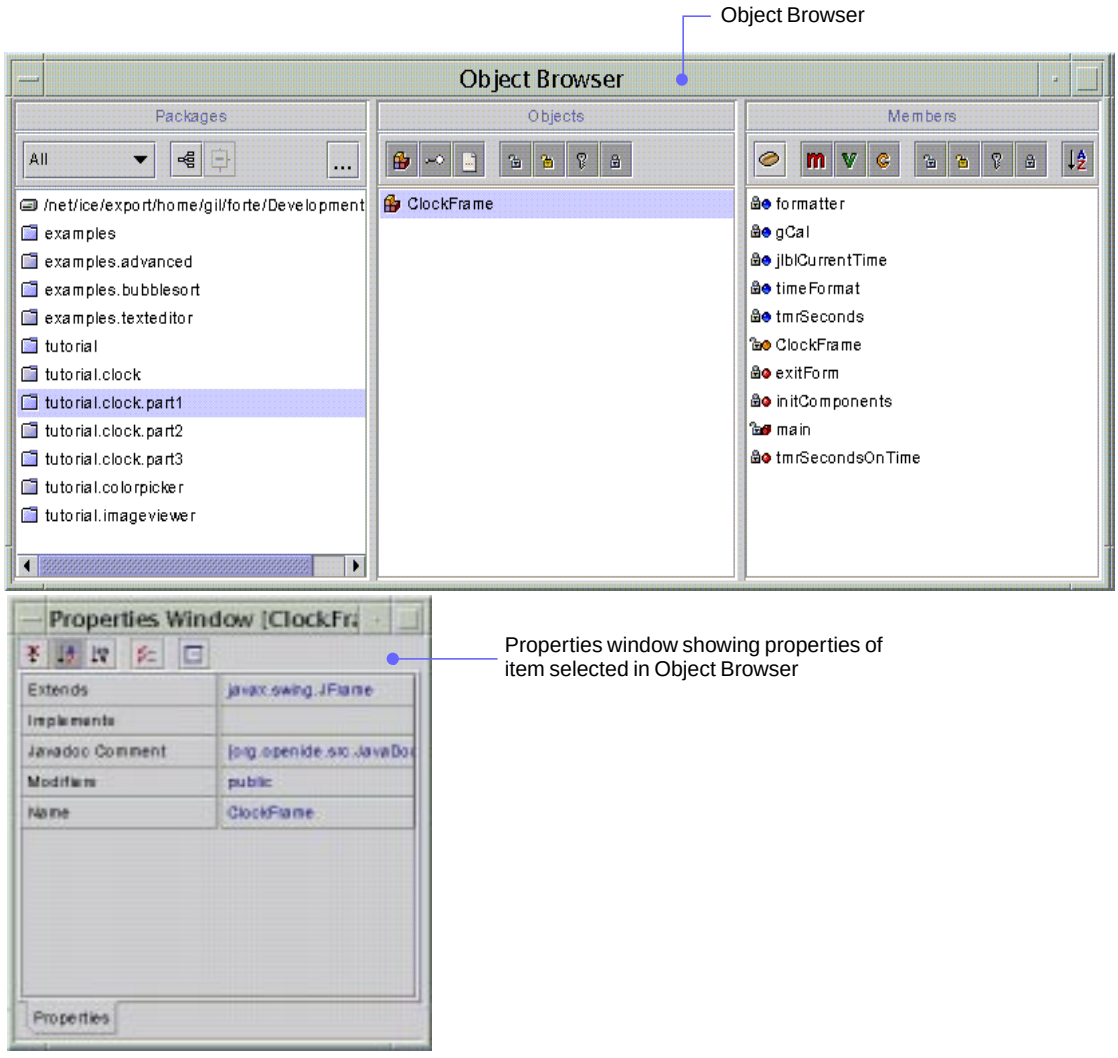
FIGURE 3 Component Inspector, Form Editor, and Editor



Browsing Workspace

The browsing workspace includes an *Object Browser* and a Properties window, as shown in the following figure. In the Object Browser, you can view the hierarchy of packages, objects (classes and interfaces), and members (methods, variables, and constructors) in your program. From the Object Browser, you can open the source code of your application by double-clicking a name in either the Objects or Members pane. The Properties window enables you to view and edit the properties of the object selected in the Object Browser.

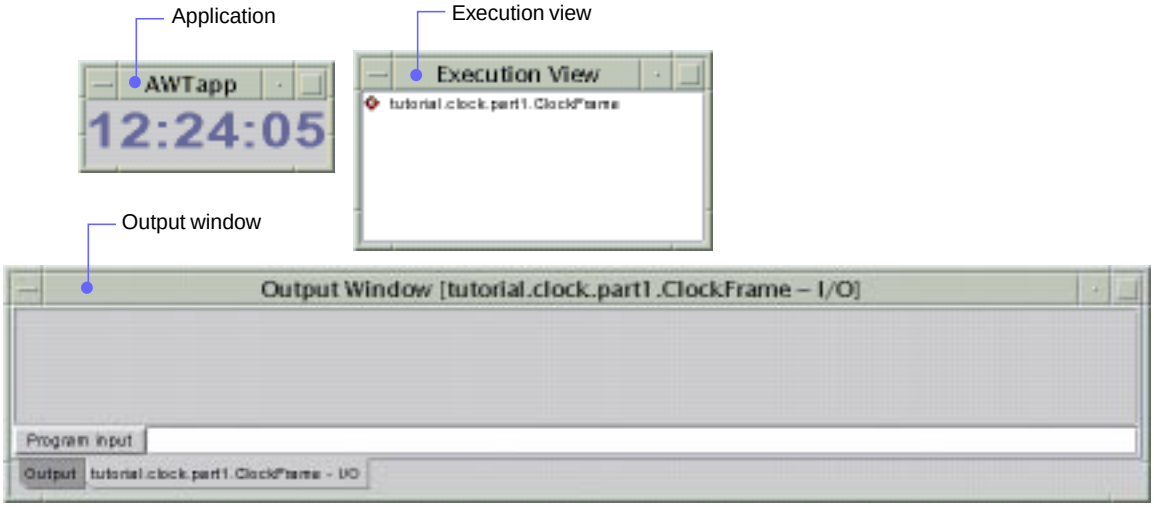
FIGURE 4 Browsing Workspace



Running Workspace

When you run your program, Forte for Java automatically switches to the running workspace, as shown in the following figure. If there are no execution errors, Forte for Java launches your application so that you can test it. If there are execution errors, Forte for Java displays them in the Output window.

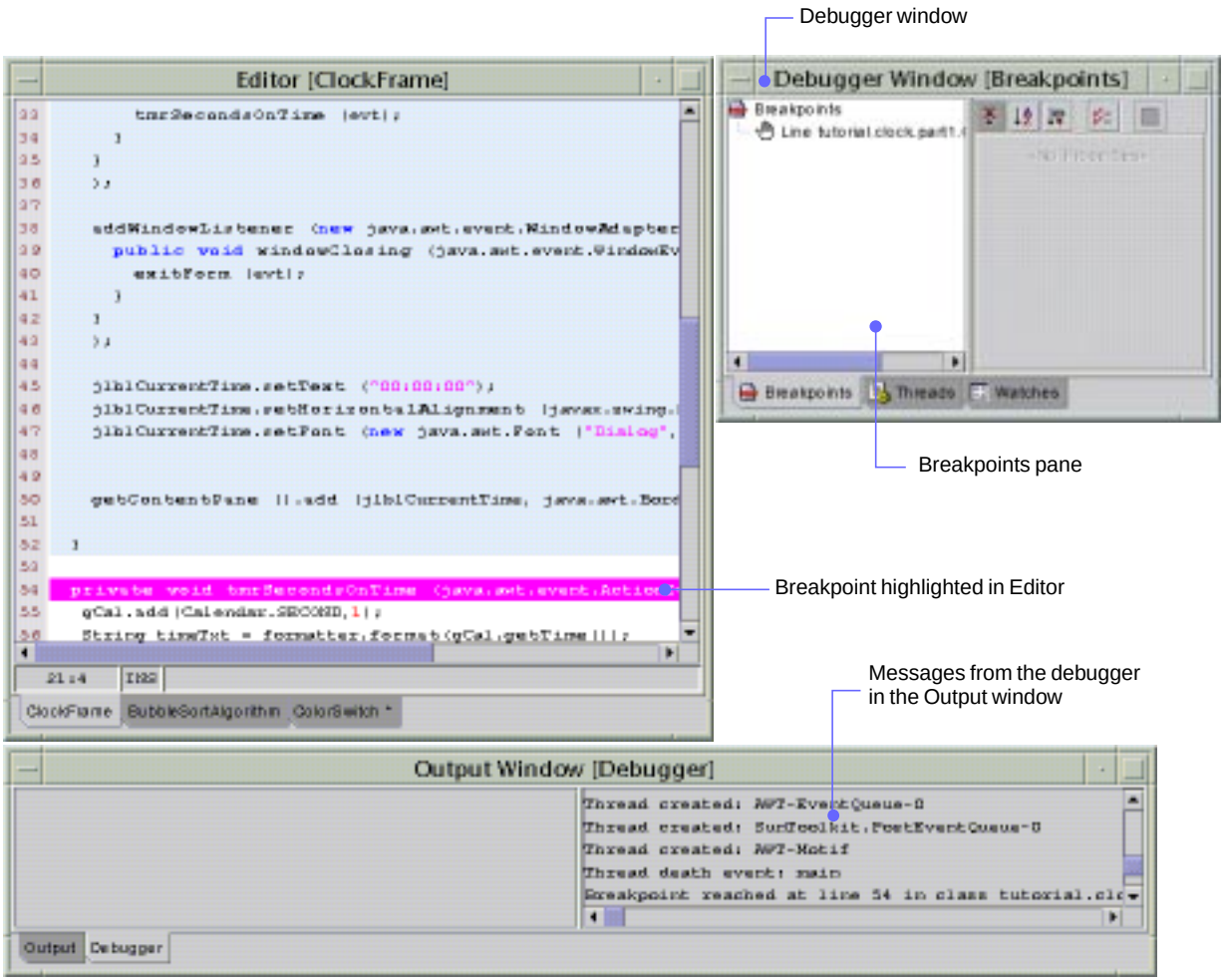
FIGURE 5 Running Workspace



Debugging Workspace

The debugging workspace includes the Debugger window and the Output window. The Debugger window has tabbed panes for setting breakpoints, monitoring threads, and watching the value of variables. The Output window displays messages from the debugger. If you have a file open, the debugging workspace also includes the Editor, which highlights breakpoints in magenta. The following figure shows the debugging workspace.

FIGURE 6 Debugging Workspace



Working With Forte for Java

This section guides you through the process of creating a Java application. You'll build a simple program that enables users to switch the color of a panel from light gray to medium gray to black.

Follow these main steps to create an application:

- Create a container from a template and place it in a project.
- Add components to your container, edit the component properties, and create connections between the components.
- Edit the Java source code.
- Compile and run the application.
- View the class hierarchy, if desired.
- Debug the application, if needed.

This "quick start" tutorial takes less than an hour to complete.

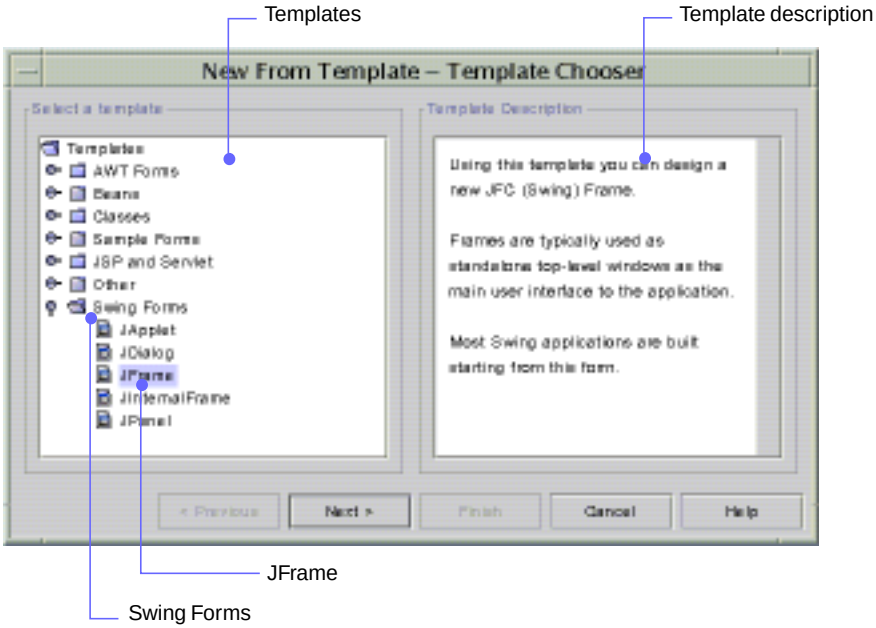
Note – Graphics in this document show Forte for Java in the Java look and feel, which is the default configuration.

Creating a Container

You'll begin working in the main window, where you will open the Template chooser and create a top-level *container*. This container will hold the other components (button and label) in your application. You'll create the container using the `JFrame` component and place it in the `colorswitch` package. (A *package* is a group of related Java classes and interfaces.)

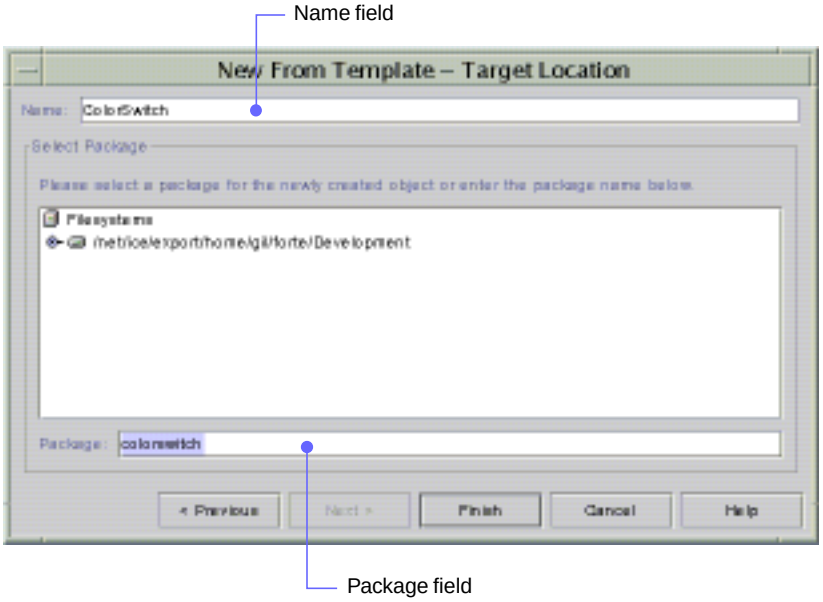
1. **From the File menu, choose New From Template.**
   The Template chooser is displayed.
2. **In the Template chooser, expand Swing Forms, then select `JFrame`.**
   A description of the `JFrame` component appears in the pane on the right, as shown in the following figure.

FIGURE 7 Template Chooser



3. **Click Next.**
   The Target Location dialog box is displayed.
4. **In the Target location dialog box, type `ColorSwitch` in the Name field and `colorswitch` in the Package field.**

FIGURE 8 Template Location Dialog Box

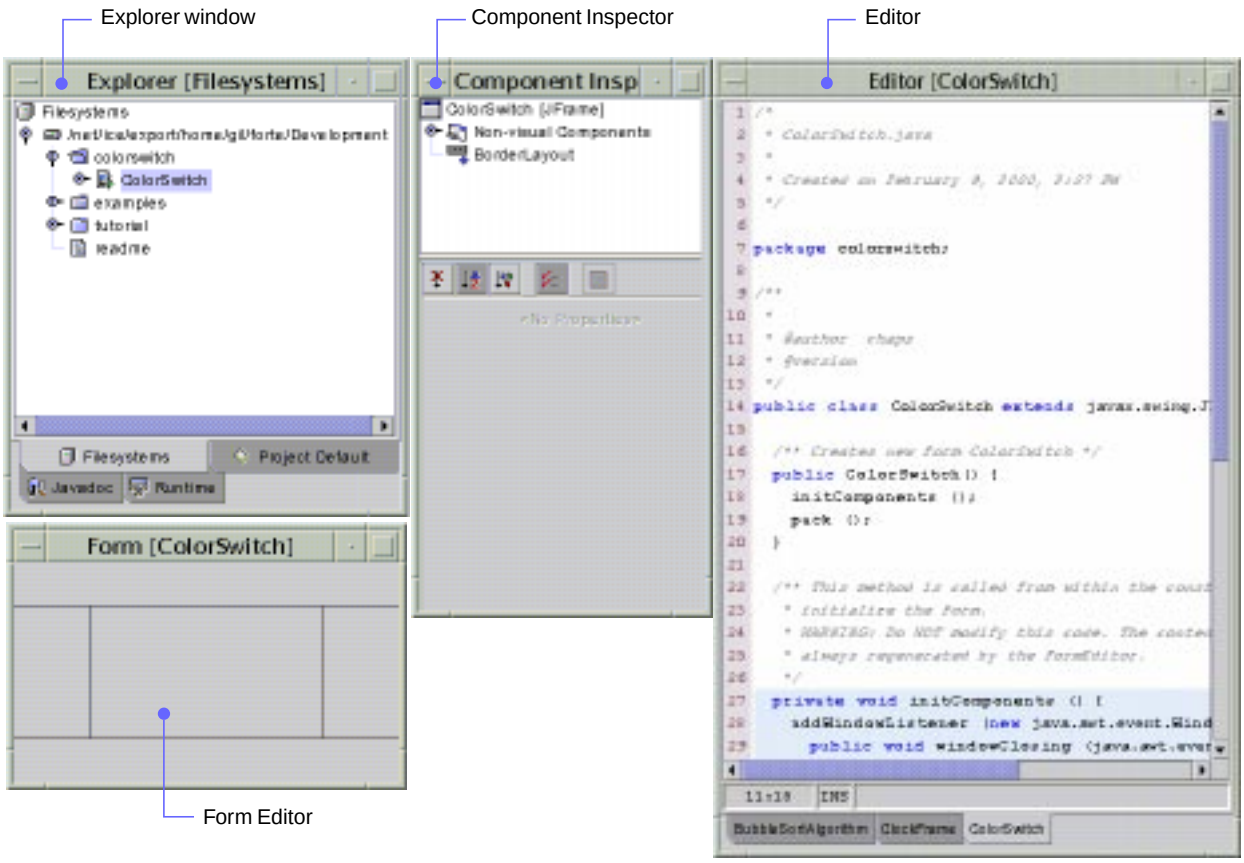


5. **Click Finish.**
   A Question alert box asks if you want to put the `ColorSwitch` object in the current project. A project organizes the files in your application into a single group, which makes them easier to find and work on.

6. **Click Yes.**
   The hierarchy in the Explorer window is expanded to show the `colorswitch` package. In addition, the Component Inspector, Form Editor, and Editor are displayed in the GUI editing workspace. The Component Inspector provides a visual representation of the components in your application and enables you to edit their properties. The Form Editor is the area into which you will add components for this container. The Editor shows the Java source code for the `JFrame` component. You can type in the white areas of the Editor only.

   The following figure shows the `colorswitch` application in the GUI editing workspace.

FIGURE 9 Application in the GUI Editing Workspace

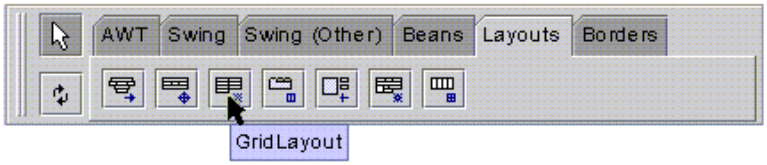


Choosing a Layout Manager

A *layout manager* assists you in determining the size and position of the components within the container. Each container type has a default layout manager. `BorderLayout` is the default for a `JFrame` component. `BorderLayout` divides the container into five sections (north, south, east, west, and center). You will switch to `GridLayout`, which creates sections equal in size and displays them in the requested numbers of rows and columns.

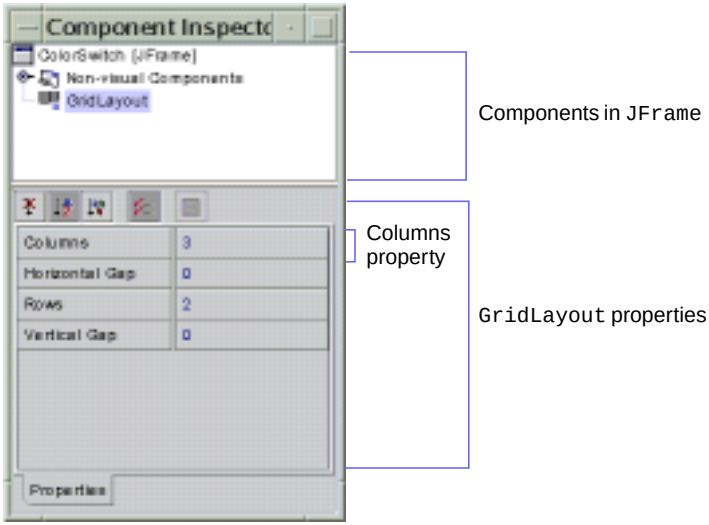
1. **In the main window, click the Layouts tab in the component palette.**
   The Layouts pane provides access to the layout managers in the *Java Foundation Classes* (JFC). When you move the pointer over a button in the toolbar, a tool tip displays the name of the layout manager.

FIGURE 10 Tool Tip for `GridLayout` in Layouts Pane



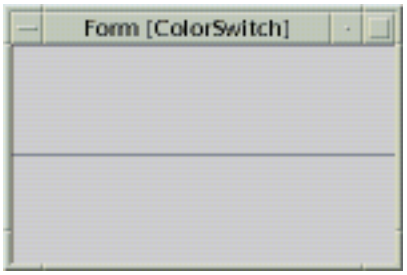
2. **In the Layouts pane, click the `GridLayout` button.**
3. **Click anywhere in the Form Editor window.**
   The Form Editor changes to display a grid three columns wide by two rows deep.
4. **In the Component Inspector window, select `GridLayout`.**
   The properties of this layout manager appear in the Properties pane of the Component Inspector.

FIGURE 11 Component Inspector for `JFrame`



5. **In the Columns property, delete 3, then type `1` and press Enter (or Return).**
   The Form Editor now displays a grid with one column and two rows, as shown in the following figure.

FIGURE 12 Form Editor With `GridLayout`

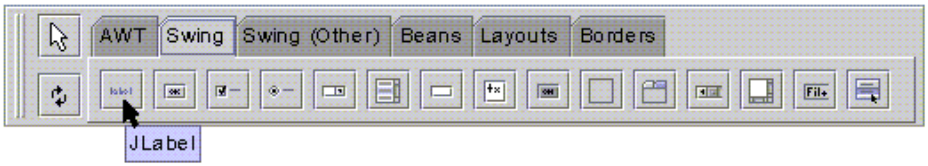


Adding a Label and Setting Its Properties

Your application uses a blank label with an opaque background to display the colors. You need to add the label and set its properties.

1. **In the component palette, click the Swing tab.**
   The Swing pane provides access to the user interface components in the Java Foundation Classes (JFC).
2. **In the Swing pane, click the `JLabel` button.**

FIGURE 13 `JLabel` in Swing Pane



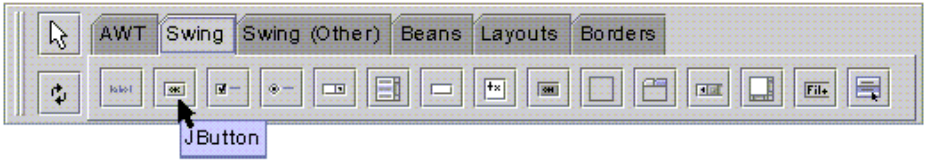
3. **Click anywhere in the Form Editor window.**
   A label with the text `jLabel1` is displayed in the Form Editor. In the Component Inspector, a node named `jLabel1 [JLabel]` is displayed (and highlighted) in the tree view and the properties for the component are displayed in the Properties pane. Source code for the label is displayed in the Editor.
4. **In the Properties pane, scroll to the opaque property, click its value, and choose `True` from the combo box.**
5. **In the text property field, delete `jLabel1` (leaving the value blank) and press Enter (or Return).**
   The Form Editor will again look like FIGURE 12.

Adding a Button

The user interacts with the Color Switch application by clicking a button, which you will now add.

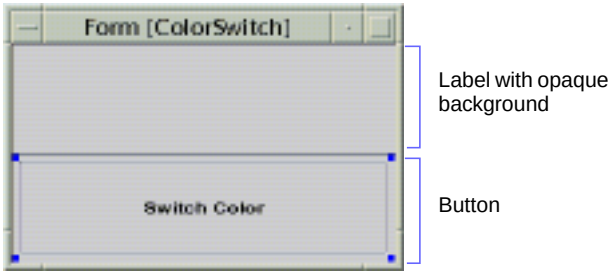
1. **In the Swing pane, click `JButton`.**

FIGURE 14 `JButton` in Swing Pane



2. **Click anywhere in the Form Editor.**
   The button is displayed in the Form Editor, its corresponding node is displayed in the Component Inspector, and its source code is displayed in the Editor.
3. **In the Properties pane, type Switch Color in the text property field, and press Enter (or Return).**
4. **Click in the font property field, and then click the ... button.**
5. **In the Property Editor dialog box, select Bold for font style and 12 for font size, and click OK.**
   The Form Editor looks as follows.

FIGURE 15 Form Editor With Label and Button

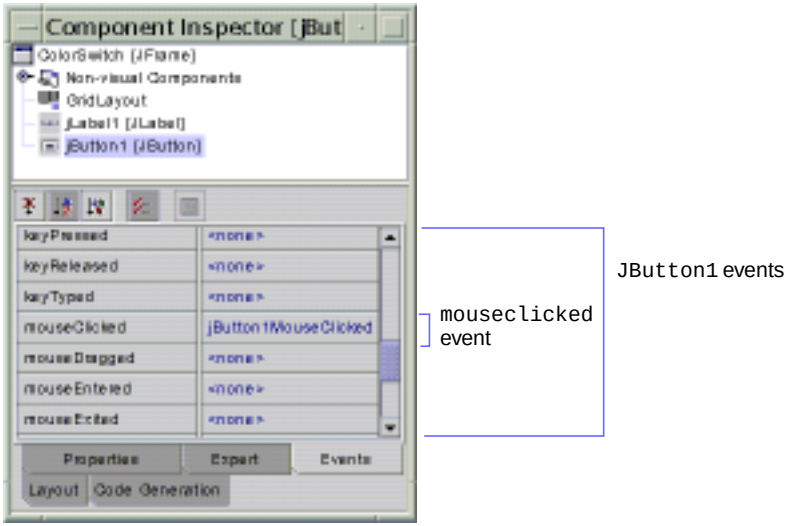


Setting Up the Button to Switch Color

In this section, you specify an event (a mouse click) to which the button can respond.

1. **In the Component Inspector window, select the `jButton1` node (if it is not already selected).**
2. **Click the Events tab.**
3. **In the Events pane, scroll to the `mouseClicked` field and click.**
   The value changes from `<none>` to `JButton1MouseClicked`.

FIGURE 16 Component Inspector Showing Events for `JButton1`



4. **Press Enter (or Return).**
   The listener code `jButton1.addMouseListener` and event method `jButton1MouseClicked()`automatically appear in the Editor, as shown in the following figure.

FIGURE 17 Editor Showing Listener Code and Event Method

Adding Event Handler Code

Now that Forte for Java has created the `jButton1MouseClicked()` method, you can add custom code for handling this event. You want a click of the button to change the color of the label.

1. **In the Editor, after the comment `//End of variables declaration` (line 77), declare a new variable:**

```
private java.awt.Color currentColor = java.awt.Color.lightGray;
```

(To use the dynamic code completion feature in the Editor, type the first few characters, choose the completed term from the list of classes, methods, and variables that the Editor displays, and press Return.)

2. **After the lines 57 and 58:**

```
private void jButton1MouseClicked (java.awt.event.MouseEvent evt) {
//Add your handling code here:
```

type the following:

```
if (currentColor == java.awt.Color.lightGray)
    currentColor = java.awt.Color.gray;
else if (currentColor == java.awt.Color.gray)
    currentColor = java.awt.Color.black;
else
    currentColor = java.awt.Color.lightGray;
jLabel1.setBackground (currentColor);
```

3. **From the File menu, choose Save.**

Compiling and Running Your Program

Now that you've created the user interface and the connections between the components, you can compile and run your application.

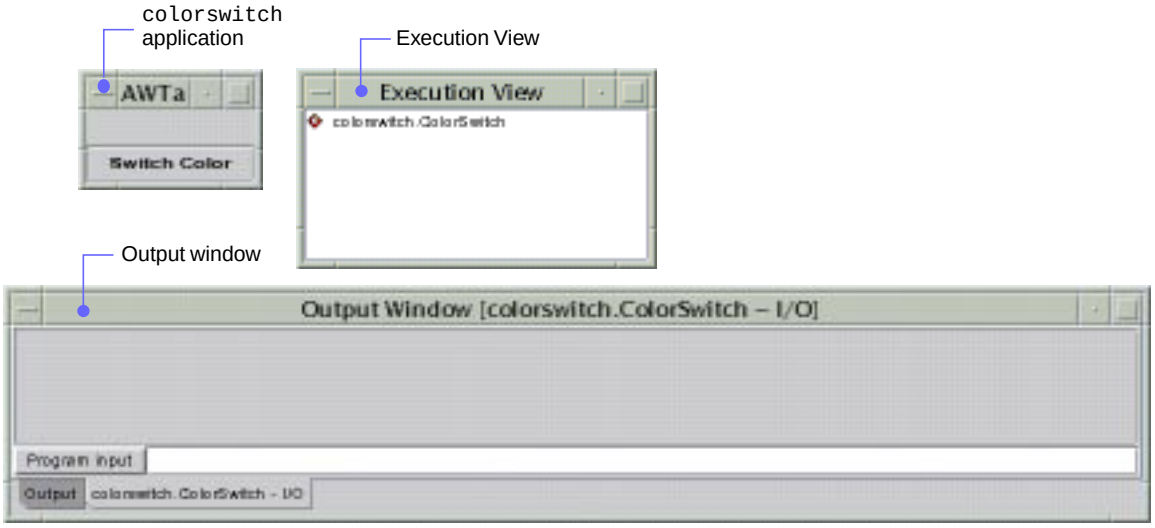
1. **From the Project menu, choose Compile Project.**
   If the build is successful, you'll see "Finished Project Default" in the status bar in the main window. If there are problems with the build, you'll see error messages in the Output window. You can double-click an error message to jump to the line in the source code where the error occurred.

   Once the build is successful, you can run the application.

2. **From the Project menu, choose Execute Project.**
   If there are no execution errors, Forte for Java switches to the running workspace and the Execution View window, the Output window, and the Select Main Class dialog box appear.

3. **In the Select Main Class dialog box, select `ColorSwitch`, then click OK.**
   The `colorswitch` application is displayed, as shown in the following figure.

FIGURE 18 Running Workspace With `colorswitch` application



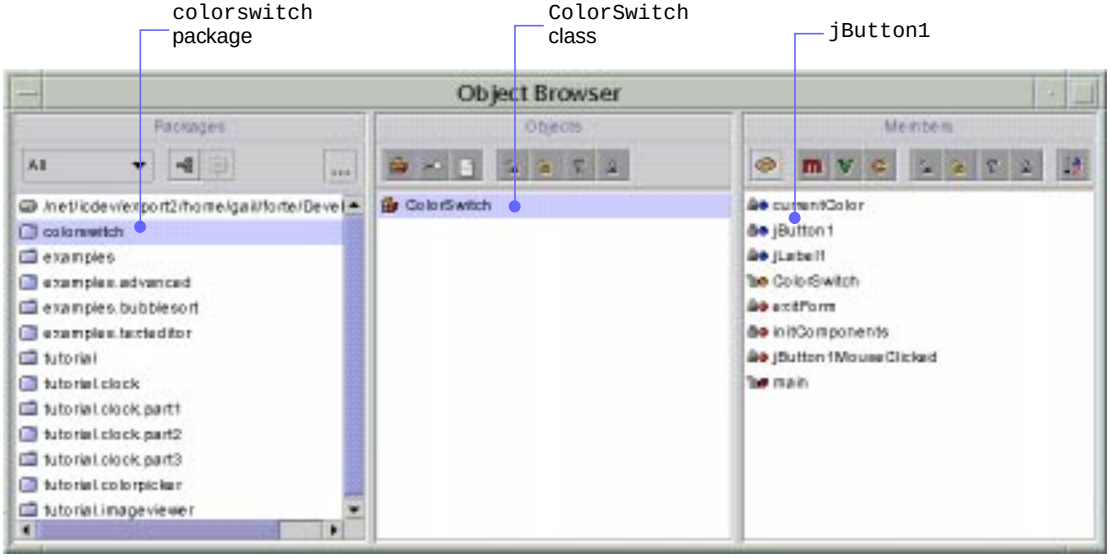
4. **In the application, click the button three times to check that the label changes from light gray to medium gray to black.**
5. **Close the application.**

Viewing the Class Hierarchy

Forte for Java has an Object Browser that enables you to view the classes, methods, and data items in your program.

1. **In the main window, click the Browsing tab.**
   The Object Browser window is divided into the Packages, Objects, and Members panes, as shown in the following figure.

FIGURE 19 Colorswitch Application in the Object Browser



2. **In the Packages pane, select the `colorswitch` package.**
   The Objects pane now shows the `ColorSwitch` class.
3. **Select the `ColorSwitch` class.**
   The Members pane shows each method and data member contained in the `ColorSwitch` class.
4. **Select `JButton1`.**
   The button's properties appear in the Properties window.
5. **Double-click the `main` method.**
   The GUI editing workspace is displayed. In the Editor, the cursor appears at the beginning of the line with the `main` method.

Debugging the Code

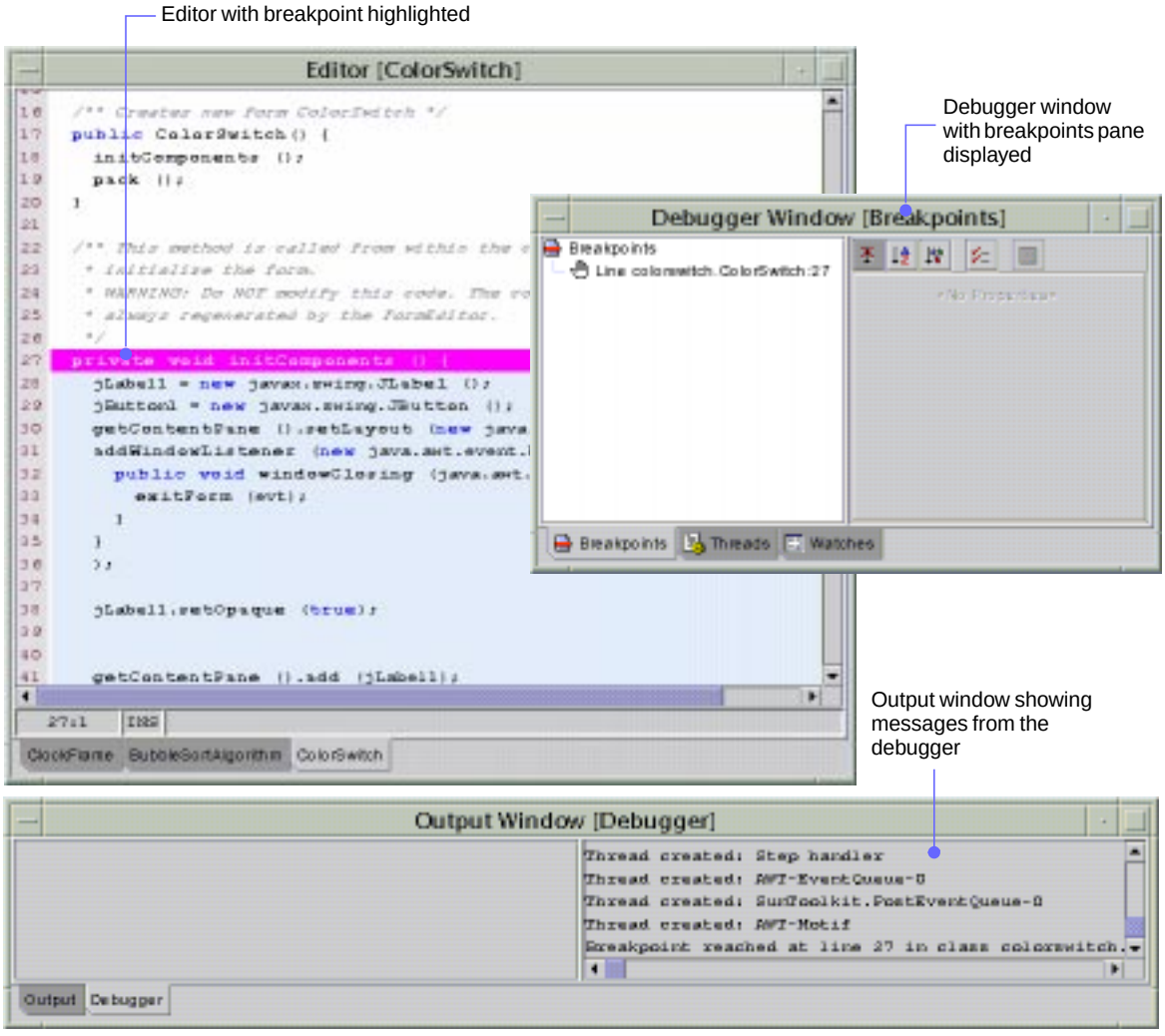
Using the debugger, you can locate and correct bugs in your program. The following steps introduce you to debugging in Forte for Java.

1. **In the GUI editing workspace, click line 27 in the Editor, which contains the `initComponents` method.**
2. **From the Debug menu, choose Add/Remove Breakpoint.**
   The line is highlighted in blue to indicate a breakpoint was set.

3. **From the Debug menu, choose Start Debugging.**
The program runs until it reaches the breakpoint. Forte for Java switches to the debugging workspace, which displays:

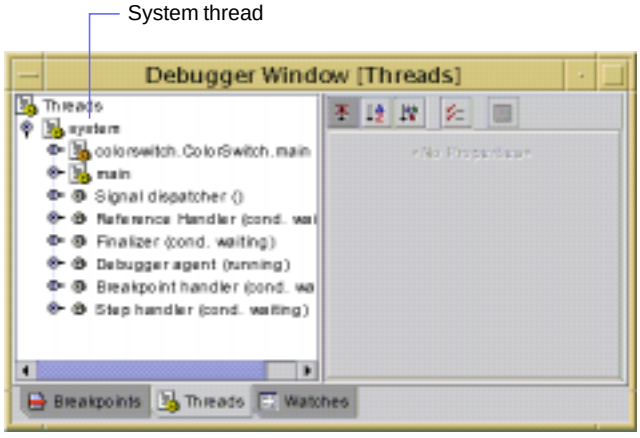
- The breakpoint line highlighted in magenta in the Editor
- Information on the breakpoint in the Debugger window
- Messages from the debugger in the Output window

FIGURE 20 Colorswitch Application Stopped at Breakpoint



4. **From the Debug menu, choose Trace Into.**
   The call to `pack` in the method `ColorSwitch` is highlighted in the Editor.
5. **From the Debug menu, choose Trace Over.**
   The next method call is highlighted.
6. **From the Debug menu, choose Trace Out.**
   The first line in the main method is highlighted.
7. **In the Editor, click the line with the `initComponents` method and then choose Add/Remove Breakpoint from the Debug menu.**
   The breakpoint is removed.
8. **In the Debugger window, click the Threads tab, then expand the system node.**
   You'll see the debugger and other threads in your program.

FIGURE 21 Threads Tabbed Pane in Debugger Window



9. **In the Debugger window, click the Watches tab.**
   You can use this pane to display the value of variables in your program.

FIGURE 22 Watches Tabbed Pane in Debugger Window

10. **From the Debug menu, choose Finish Debugging.**

Taking Best Advantage of Forte for Java

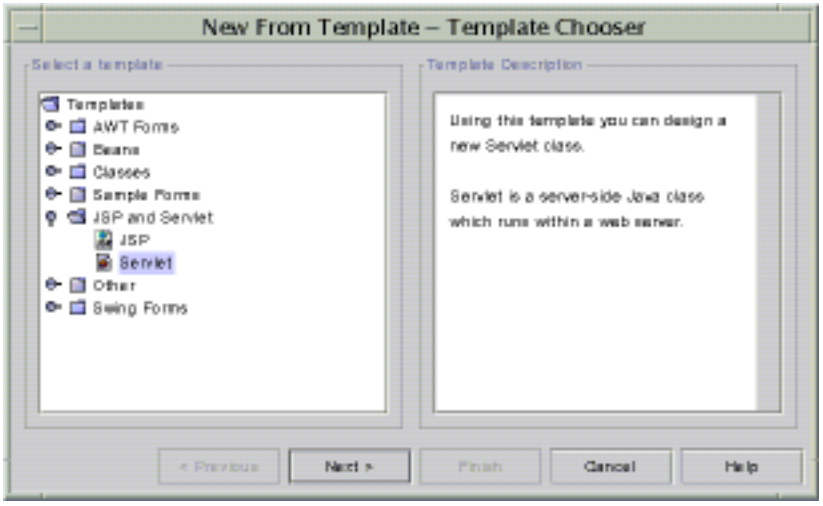
The previous tour of Forte for Java introduced you to the main interface and the steps involved in creating an application. Following are some additional features of the environment that enable you to create applets and applications more efficiently.

Using Templates

Templates are a powerful tool in the Forte for Java environment. When you create an object, you use its *template*, which determines the initial appearance and behavior of the object. Templates can reduce the amount of time and effort involved in creating your application.

Java components, such as Swing and AWT containers, are provided as standard templates. Forte for Java also provides templates for applets, classes, dialog boxes, HTML files, text files, and bookmarks. To access the Template chooser (shown in the following figure), choose New From Template from the File menu.

FIGURE 23 Template Chooser

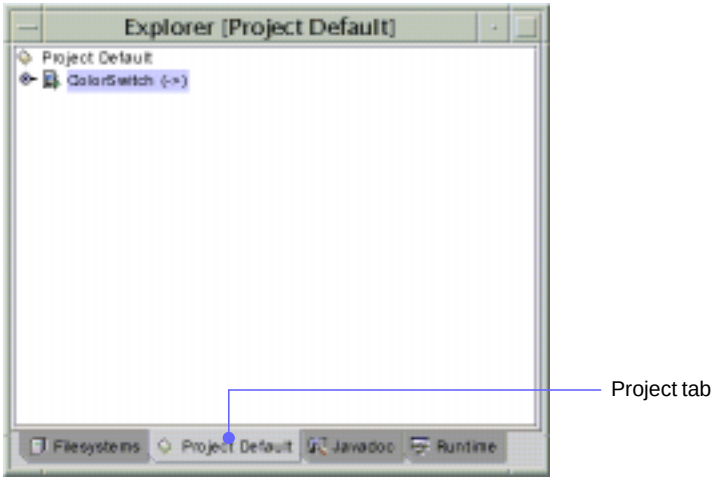


Using Projects

A *project* enables you to organize the files required to produce an applet or an application. When you organize your files into a project, you can operate on them as a whole. For example, when you compile a project, you compile all of the Java source files in it.

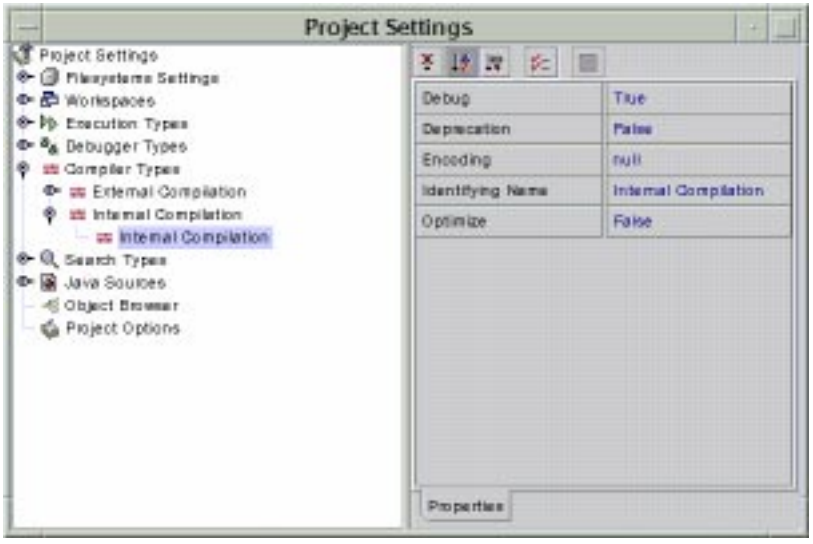
You create and manage projects using the Project menu in the main window. To view the files in the current project, use the Project tab in the Explorer window, as shown in the following figure.

FIGURE 24 Project Tab in Explorer Window



For each project, you can specify a set of attributes, such as which compiler and debugger types to use. You set attributes in the Project Settings dialog box (shown in the following figure), which is available from the Projects menu.

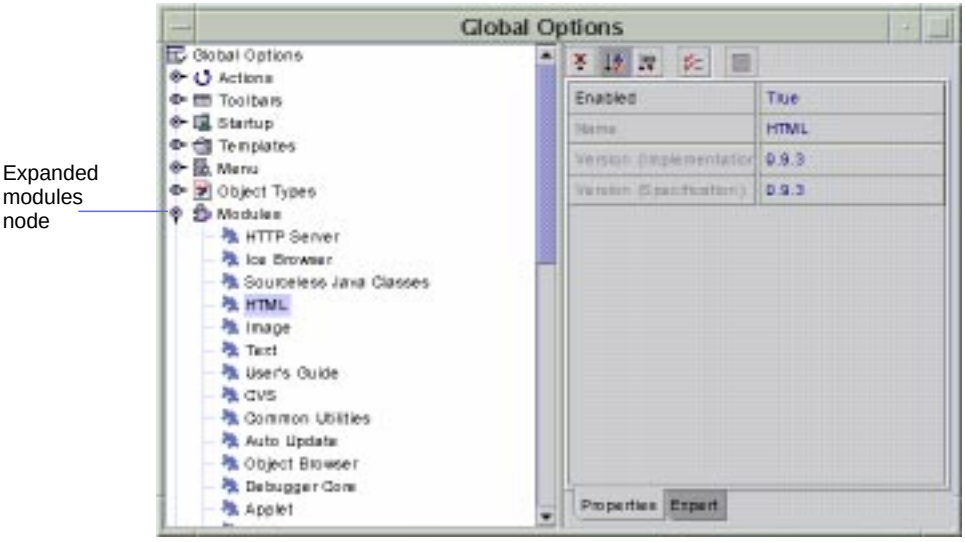
FIGURE 25 Project Settings Window



Using Modules

Forte for Java is built entirely from modules, independent pieces of software that are compiled separately. Even features central to the environment, such as the Editor, Debugger, and Form Editor, are modules. You can expand this modular structure with plug-in extensions from Sun Microsystems or third parties. To view the modules currently installed in your environment, choose Global Options from the Tools menu. When the Global Options window opens, expand the Modules node, as shown in the following figure.

FIGURE 26 Global Options Window With Modules Node Expanded



To install a new module, choose Update Center from the Help menu and follow the directions in the dialog box.

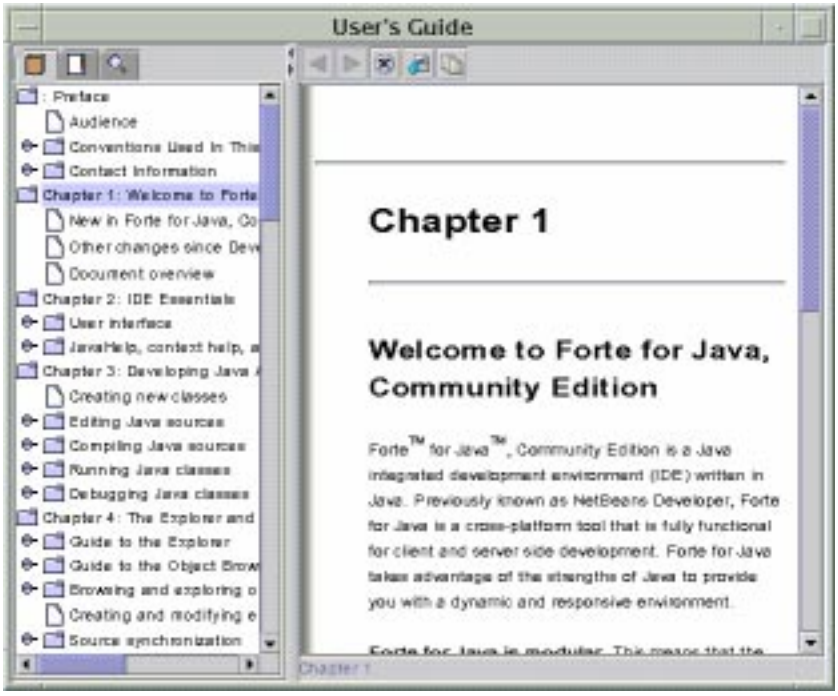
Customizing Your Work Environment

Forte for Java lets you customize your work environment in a number of ways. You can create, delete, and change menus, toolbars, and workspaces. You can also add an object to the component palette and then use that object in your applets and applications. You can look at files in the Editor side-by-side, instead of in separate tabbed panes, and you can change the Editor's keyboard shortcut assignments. You can make these changes (and change other aspects of your environment) in the Global Options window, shown in the preceding figure. You access the Global Options window from the Tools menu.

Getting More Information

The *Forte for Java (Community Edition) User's Guide* describes both conceptual information and how to use Forte for Java. You can view this document online by opening the Help menu and then choosing Documentation. In addition, you can press F1 in most windows to open the user's guide to information specific to the task you are performing. The following figure shows a page from the user's guide.

FIGURE 27 User's Guide Displayed in Help Window



Forte for Java also includes four online tutorials, which introduce you to additional features of the environment, such as how to build a JavaBeans™ component architecture. To access the tutorials, open the Help menu, and choose Tutorial.

From the Help menu, you can also open a web browser and access the Forte for Java web site (www.netbeans.com) by choosing Forte for Java Home on the Web. This site provides installation instructions, module updates, and other information.

Glossary

Abstract Window Toolkit	An API that provides graphical user interfaces for Java programs. The Abstract Window Toolkit (AWT) also provides imaging tools, event-handling methods, layout managers, and data transfer classes. The AWT components are implemented using native-platform versions of the components and have largely been replaced by the Swing components, which have a pluggable look and feel.
applet	A program, written in the Java programming language, that runs in a web browser.
application	A standalone software program that enables the user to perform a specific task, such as database management.
class	A group of attributes and methods that define the implementation of a particular type of object.
component	An object that is identified by its properties, operations, and relationships. For example, a button is a visual component whose properties include size and foreground and background color.
Component Inspector	A window in which you can view both the visual (such as a button) and non-visual (such as a layout manager) components in your application. From the Component Inspector, you can modify a component's properties and specify its events.
component palette	A collection of toolbars that provides easy access to frequently used components, including AWT and Swing components. You can create your user interface by clicking a component in the component palette and then clicking in the Form Editor.
container	A component that contains other components. Windows and dialog boxes are examples of top-level containers. Panels, scroll panes, and tabbed panes are examples of intermediate-level containers.
dock	To anchor an object, such as a toolbar, to the edge of the window or pane to which it applies.
dynamic code completion	The automatic completion of an expression that you are typing in the Editor window. To use dynamic code completion, type the first few characters of the expression, and then press CTRL+SPACE. A list of of classes, methods, and variables that can be used to complete the expression is displayed.

Editor A tool for editing Java, HTML, and plain text files as well as files specified by modules. Source code in the Editor window is automatically updated and generated as you work in the Form Editor window and Component Inspector. The generated source code is indicated by a shaded background and cannot be edited in the Editor window.

event An action to which an object can respond. Most events are initiated by a user action, such as a click, key press, or mouse movement.

event handler A method that is called when an event is triggered on a component.

Explorer window A window that provides a unified view of all objects and files in the Forte for Java environment. The Explorer window is a good starting point for working with your application, including organizing files, editing object properties, and creating component connections.

Form Editor window An area for creating and modifying a graphical interface. You can select a component (such as a panel, scroll bar, or menu) in the component palette and add it to your graphical interface by clicking in the Form Editor window.

Java Foundation Classes An extension to the Abstract Window Toolkit (AWT) that provides the Swing classes, a collection of graphical user interface components with a pluggable look and feel. The Java Foundation Classes (JFC) also provide the Java Accessibility API, which can be used to create applications that interact with assistive technologies. For the Java 2 platform, the JFC also includes the Java 2D API (for 2D graphics and imaging) and drag and drop.

layout manager A property of a container component that controls the size and location of components within the container. A layout manager ensures that the container can adjust to resizing and to differences between systems, such as different font sizes. The Java platform supplies six layout managers: `BorderLayout`, `BoxLayout`, `CardLayout`, `FlowLayout`, `GridBagLayout`, and `GridLayout`.

main window A window that acts as the control center for Forte for Java. The main window contains menus, toolbars, and a component palette for developing Java applets and applications. From the main window you can access the five workspaces in Forte for Java.

method A procedure that belongs to a class and that can be applied to a specific object or the class itself.

module An independent piece of software that is part of a larger program but is usually compiled separately. Modules are implemented in such a way that you can change one module without affecting the other modules in the program.

Object Browser A three-pane window in which you can view the hierarchy of packages, objects, and members in your application. Like the Explorer window, you can use the Object Browser as a base for many tasks in the development of your application. For example, you can open a source file and add new packages, objects, and members from the Object Browser.

package A collection of Java classes and interfaces, grouped in a single entity.

project A collection of files that make up an applet or application. The files in a project can be operated on as a whole.

property An attribute or characteristic of a GUI object that you can set. The properties of an object might define its size, color, and value.

Swing components A collection of GUI components with a pluggable look and feel so you can design an application that can have the look and feel of any OS platform. Swing is part of the Java Foundation Classes and includes interface elements such as windows, dialog boxes, choosers, panels, panes, menus, controls, text components, tables, lists, and tree views.

template Software code that serves as a guide for creating a component. A template provides the initial appearance and behavior of the object, which you can easily change. In the Forte for Java environment, components such as Swing and AWT containers are provided as templates.

workspace A collection of windows with related functions. For example, when you edit your user interface, you use a workspace that displays the Component Inspector, the Form Editor, and the source Editor. When you debug your program, you use a workspace that displays a window for setting breakpoints, monitoring threads, and watching the value of variables.

Index

V

W